# AUTOMATED TEST CASES GENERATION FOR OBJECT ORIENTED SOFTWARE

A.V.K.SHANTHI

Research Scholar, Sathyabama University,
Chennai-119, Tamil Nadu, India,
avks_15@yahoo.com.

DR.G.MOHANKUMAR

Principal ,Park College of Engineering,
Coimbatore, Tamil Nadu, India,
gmohankumar68@yahoo.com

**Abstract**

Software testing is an important activity in software development life cycle. Testing includes executing a program on a set of test cases and comparing the actual results with the expected results. To test a system, the implementation must be understood first, which can be done by creating a suitable model of the system. UML is widely accepted and used by industry for modeling and design of software systems. A novel method to automatically generate test cases based on UML Class diagrams guaranteed in many ways. Test case generation from design specifications has the added advantage of allowing test cases to be available early in the software development cycle, thereby making test planning more effective. Here is a technique in which a new approach using data mining concepts is designed and that algorithm is to be used to generate test cases. The Tool generates a novel automated test case that is much superior, less complex and easier to implement in any Testing system. Where in this Tool, information from UML Class diagram extracted and mapped, tree structure is formed with help of those information's, Genetic Algorithm implemented as data mining technique, where Genetic crossover operator applied to discover all patterns and Depth First Search algorithm implement to Binary tree's formed to represent the knowledge i.e., test cases. Path coverage criterion is an important concept to be considered in test case generation is concern. This paper presents valid test cases generation scheme which is fully automated, and the generated test cases to satisfy transition path coverage criteria.

**Keywords: Test Cases, Class Diagram, UML (Unified Modeling Language), Data Mining, Genetic Algorithm, Binary Tree, Depth First Search.**

## 1. Introduction

Software testing is an important activity in software development life cycle. Software organizations spend considerable portion of their budget in testing related activities. A well tested software system will be validated by the customer before acceptance. Testing includes executing a program on a set of test cases and comparing the actual results with the expected results. Testing should also focus on fault prevention. Test cases are usually derived from software artifacts such as specifications, design or the implementation. To test a system, the implementation must be understood first which can be done by creating a suitable model of the system. A common source for tests is the program code. Every time the program is executed, the program is tested by the user. So we have to execute the program with the specific intent of fixing and removing the errors. In order to find the highest possible number of errors, tests must be conducted systematically and test cases must be designed using disciplined techniques.

UML [Unified Modeling Language] is a widely accepted set of notations for modeling object oriented system. It has various diagrams for depicting the dynamic behavior of objects in a system. We use Class diagrams represented in the form of a tree to extract test cases to verify/validate the behavior of class concerned. Evolutionary Genetic algorithm is used to generate all possible valid test cases for the software to be tested or developed. Data is obtained from UML design i.e., class diagram. A Novel method is proposed to form tree structure among class in design, based on their relationships. Depth First Searching (DFS) algorithm is used to generate test cases from the binary tree formed. All requirements of data mining concept implemented in very established way to produce optimal number test cases. Data Mining is the non-trival process of identifying valid, novel, potentially useful ultimately understandable patterns of data.
Data mining techniques support automatic exploration of data and attempts to source out patterns and trends in the data and also infer rules from these patterns which will help the user to support review and examine decisions in some related business or scientific area.

**2. Objective**

New model based approach for automated generation of test cases in object oriented systems has been analyzed. The test cases are derived by analyzing the dynamic behavior of the class due to internal and external stimuli. The UML and its diagrams are widely used to visually depict the static structure and more importantly for us, the dynamic behavior of such applications gives developers and testers the capability of automatically generating black-box conformance tests early on. The scope of the Tool has been limited to the class diagrams taken from the Unified Modeling Language model of the system. Class diagrams are the backbone of almost every object oriented method, including UML. They describe the static structure of a system.Evolutionary Genetic Algorithm and Genetic Algorithm's tree crossover has been proposed to bring out all possible test cases of a given class diagram. Illustrative case study has been implemented to establish the effectiveness of our methodology coupled with mutation analysis.

Specification-based testing uses information derived from a specification to assist testing as well as to develop program. Testing activities consist of designing test cases that are a sequence of inputs, executing the program with test cases, and examining the results produced by this execution. Testing can be created earlier in the development process so the developer will often find inconsistency and ambiguity in the specification and so the specification can be improved before the program is written.

**3. Methodology**

Construct a class diagram for any software to developed using Rational rose software. UML Model file will be saving with an extension .MDL. Write Parser in java to extract all possible information from file. From the extracted information group the attributes and methods for their respective class and also look for the relationship between the classes to form a tree structure. If a class contains more than one aggregated, associated, specialized, dependent class draw tree structure for one set of associated class then move to other such class for forming new tree. Tree structure can be formed such way class name occupies in next to start node and left side with its attributes and right with its method. Apply Genetic Algorithm two point cross over randomly from the obtained trees. Apply to all pair of trees formed. Evolutionary GA used to obtain the fittest tree. Since the tree structure is not in the form of binary, covert the obtained tree in form of binary tree by next class of present class in tree is reconstructed to occupy its last method present in its present class. Then apply Depth First Searching algorithm to all binary trees formed. To generate all possible and valid test cases. Efficiency of test cases can be analyzed with help of Mutation Testing.

- Tree Formation
- New Trees Formation

- Test Cases Generation
- Test cases Evaluation

**4. Existing Testing Technique**

- Test Cases are generated with help of Object, Sequence, Activity, Collaboration, State-Chart diagrams. Numerous applications developed with help activity diagram to generate test cases.
- Lacking in generalization and automation. These are two basic pit falls of present procedure to software testing. Basically testing done manually even for test cases generation. When it is being automated it fails to be generalized only for that particular or some application it fits.
- Validation of obtained test cases is manual. In case automated system obtained test cases should be validated manually to use such test cases to analysis the software.
- Test Cases are generated based on code by which the application being developed.
- Existing system based on code analysis certainly depend on language used and application. Generalization is quite difficult.

**5. Proposed Testing Technique**

- Since test case generation from design specifications has the added advantage of allowing test cases to be available early in the software development cycle, UML Class diagram is used.
- Implementation of data mining technique to generate optimal test cases.
- Validation of test cases is automated. Evolutionary Genetic algorithm is used generate valid and optimal patterns necessary.
- Test cases can be easily generated in case of regression testing. Even in case of reengineering of software, updates once reflected in design specification automatically new set test cases can be generated.
- Reduce complexity to analysis the code.
- Specification-based testing uses information derived from a specification to assist testing as well as to develop program. Testing can be created earlier in the development process so the developer will often find inconsistency and ambiguity in the specification.

**6. Mutation Analysis**

To analysis efficiency of test cases generation mutation process is invoke. Mutation is process of injecting faults to software to analysis the software efficiency. Mutation testing is a process by which faults are injected into the system to verify the efficiency of the test cases. mutation testing had reduced its practical use as a method of software testing, but the increased use of object oriented programming languages and unit testing frameworks has led to the creation of mutation testing tools for many programming languages as a means to test individual portions of an application.

Table 1: Mutation Analysis

| OPERATOR | FAULTS INJECTED | FAULTS FOUND |
|---|---|---|
| Function | 10 | 10 |
| Data Name | 10 | 10 |
| Data Member | 10 | 10 |
| Sub Class | 8 | 8 |
| Total | 38 | 38 |

Analysis based on above operators our proposed identify 100% bugs based on obtained test cases and path can be analyzed based on obtained test cases.

## 7. Conclusion

In recent trend Model-Based test case attracts many researchers by using some data mining concept to produce an automated optimal test case. By which human and cost effort are minimized. Since test case generation from design specifications has the added advantage of allowing test cases to be available early in the software development cycle, UML Class diagram is used. Our approach using class diagram in UML Model, Comparatively with Evolutionary Genetic Algorithm yields optimal valid test cases than with only genetic crossover operator, after Applying Depth First Searching algorithm.

There are various advantages: Specification-based testing uses information derived from a specification to assist testing as well as to develop program. Evolutionary Genetic Algorithm has been proposed and implemented to bring out all possible valid test cases of a given class diagram. Since test cases are obtained is valid one so it is not mandatory to evaluate manually.

## Reference


[1] M.Prasanna, S.N.Sivanandam, Venkatesan, R.Sundarrajan,15, 2005,"A SURVEY ON AUTOMATIC TEST CASE GENERATION", Academic Open Internet Journal.

[2] Baikuntha Narayan Biswal, Pragyan Nanda, Durga Prasad Mohapatra, 2008 IEEE, "A Novel Approach for Scenario-Based Test Case Generation",International Conference on Information Technology.

[3] Chang-ai Sun, 2008 IEEE, "Transformation-based Approach to Generating Scenario-oriented Test Cases from UML Activity Diagrams for Concurrent Applications", Annual IEEE International Computer Software and Applications Conference.

[4] Bin Lei, Linzhang Wang, "Xuandong Li, UML Activity Diagram Based Testing of Java Concurrent Programs for Data Race and Inconsistency ", 2008 International Conference on Software Testing, Verification, and Validation.

[5] P. Samuel, R. Mall, A.K. Bothra,2008 "Automatic test case generation using unified modeling language (UML) state diagrams ",Published in IET Software.

[6] Emanuela G. Cartaxo, Francisco G. O. Neto and Patr´ıcia D. L. Machado, "Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems", IEEE 2007.

[7] Hyungchoul Kim, Sungwon Kang, Jongmoon Baik, Inyoung Ko, "Test Cases Generation from UML Activity Diagrams ", Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing.

[8] Supaporn Kansomkeat and Sanchai Rivepiboon, "Automated-Generating Test Case Using UML Statechart Diagrams ",SAICSIT 2003.

[9] Santosh Kumar Swain, Durga Prasad Mohapatra, and Rajib Mall, "Test Case Generation Based on Use case and Sequence Diagram", Int.J. of Software Engineering, IJSE Vol.3 No.2 July 2010

[10] P. McMinn and M. Holcombe. Evolutionary testing of statebased programs. In *GECCO*, pages 1013–1020, 2005.

[11] B. Meyer. Design by contract. *IEEE Computer*, 25(10):40– 51, 1992.

[12] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, London, UK, 1996.

[13] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[14] J. Offutt and A. Abdurazik. Generating tests from UML specifications. In *UML*, pages 416–429, 1999.

[15] J. Offutt, S. Liu, A. Abdurazik, and P. Ammann. Generating test data from state-based specifications. *Softw. Test., Verif. Reliab.*, 13(1):25–53, 2003.

[16] R. P. Pargas, M. J. Harrold, and R. Peck. Test-data generation using genetic algorithms. *Softw. Test., Verif. Reliab.*, 9(4):263–282, 1999.

[17] P. Tonella. Evolutionary testing of classes. In *ISSTA*, pages 119–128, 2004.

[18] N. Tracey, J. Clark, and K. Mander. Automated program flaw finding using simulated annealing. In *ISSTA '98*, pages 73–81. ACM Press, 1998.

[19] N. Tracey, J. A. Clark, K. Mander, and J. A. McDermid. An automated framework for structural test-data generation. In *ASE*, pages 285–288, 1998.

[20] N. J. Tracey. *A search-based automated test-data generation framework for safety-critical software*. PhD thesis, University of York, 2000.